САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Рубежанская Владислава Сергеевна

Выпускная квалификационная работа бакалавра

Рекомендательная система для образовательного контента

Уровень образования: Бакалавриат

Код 1.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5005.2017 «Прикладная математика, фундаментальная информатика и программирование»

Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:

Старший преподаватель

Уланов А. В.

Рецензент:

к.ф. – м.н.

Главный администратор CRIS-системы Pure СПбГУ

Лепихин Т.А.

Санкт-Петербург

2021

Содержание

Введение в проблематику

В книге «Длинный хвост» Крис Андерсон сказал: «Мы выходим из эпохи информации и вступаем в эпоху рекомендаций» [1]. Огромный объем информации окружает людей, что помогает им принимать более правильные решения. Однако качество таких решений снижается от переизбытка знаний. В настоящее время новые технологии и быстрый рост Интернета упростили доступ к информации для всех категорий людей, поставив перед образованием новые совершенно новые задачи. Как следствие пандемии сейчас активно развивается онлайн-образование. Все больше и больше учебных заведений регулярно используют компьютеризированные инструменты для обучения и тестирования, собираются огромные объемы данных. С развитием технологий появляются новые возможности и потребности в образовании, такие как индивидуальный подход, объективность оценки, встают вопросы, как лучше направить студентов в процессе обучения среди такого большого разнообразия книг, курсов и упражнений. Одним из самых больших препятствий в современных методах обучения является то, что всем учащимся предоставляется один и тот же учебный план и учебные материалы. Однако студенты обладают разными навыками и поступают на учебу с разными знаниями. Учителя должны иметь возможность создавать полностью индивидуализированные образовательные программы, основанные на успеваемости и трудностях обучения каждого ученика, так же, как и студентам нужно предоставить качественные инструменты, помогающие ориентироваться в таком многообразии учебных материалов. Чтобы удовлетворить эту потребность, было разработано множество различных информационных и рекомендательных стратегий. Системы рекомендаций - одна из них. Системы рекомендаций пытаются помочь пользователю, представляя те объекты, которые могут быть более интересны, исходя из его известных предпочтений или предпочтений других пользователей с аналогичными характеристиками.

Постановка задачи

Целью этой работы является исследование различных подходов к построению рекомендательных систем (РС) и в первую очередь разработка рекомендательной системы, которая основываясь на введенном идентификаторе, определяет тематику запроса и рекомендует существующие онлайн-курсы. Также, разработка РС, которая опираясь на данные тестирования/контрольной работы, может использоваться в образовательном контенте для повышения качества обучения.

Для достижения поставленной цели были поставлены следующие задачи:

1. Изучение рекомендательных систем и алгоритмов
2. Изучение существующих библиотек в Python, необходимых для разработки рекомендательной системы
3. Обзор существующих решений в задачах построения рекомендательных систем
4. Проектирование и реализация рекомендательной системы
5. Анализ полученных рекомендаций с целью определения успешности выбранного подхода.

Данная работа разбита на 2 части. В первой рассматривается теория рекомендательных систем: основные виды и алгоритмы. Во второй части рассматривается подход к реализации рекомендательной системы, которую можно использовать в образовании и анализ проведенных экспериментов и всему с ними связанному.

Обзор литературы

Область применения рекомендательных систем обширна. Данные системы позволяют поддерживать «диалог» с пользователем в автоматическом режиме, выбирая для него наиболее подходящий контент в зависимости от выбранного метода выбора рекомендаций. Таким образом, данные системы стали привлекательны не только в развлекательной и коммерческой области, но так и применяться в образовательных целях [11] [12].

В статье [13] авторы демонстрируют модель компьютеризированной адаптивной практики и мониторинга. Эта модель используется в Maths Garden, веб-системе мониторинга, которая включает в себя сложную веб-среду, в которой дети могут заниматься арифметикой. Используя модель ответа на вопросы, основанную на рейтинговой системе Elo (1978) и явном правиле подсчета баллов, оценки способностей людей и сложности вопросов обновляются с каждым ответом, что позволяет проводить калибровку элемента очень быстро.

В работе [11] была разработана адаптивная, краудсорсинговая, веб-платформа, ориентированная на студентов. В зависимости от уровня знаний и потребностей в обучении каждому учащемуся рекомендуется задать набор индивидуальных вопросов. Для студентов, которые заинтересованы в предоставлении поддержки в обучении или поиске партнеров по обучению, платформа рекомендует сеансы взаимного обучения на основе их доступности, уровня знаний и предпочтений.

Статья [18] демонстрирует концепции электронного обучения, систем рекомендаций и глубокого обучения. В ней рассматриваются основные парадигмы рекомендательных систем с использованием явной и неявной обратной связи, а также различные методологии, которые были реализованы для разработки рекомендательных систем для улучшения обучения.

В исследовании [19] предлагается рекомендательная модель курсов, основанная на совместной фильтрации, чтобы помочь студентам и академическим консультантам во время регистрации на курс. Применяется совместная фильтрация, поскольку она дает рекомендации, основанные на успеваемости учащихся по предыдущим предметам. Мерой сходства в данном исследовании была выбрана корреляция Пирсона.

Не осталось в стороне от этой темы и российское интернет сообщество. Появляются все больше онлайн-курсов и электронный лекций. [14] Они кратко описывают, как работают рекомендательные системы, разбирают простейшие алгоритмы и метрики, а также исследуют возможность измерения качества рекомендаций и направление развития.

Глава 1. Рекомендательные системы

1.1 Определение

Система рекомендаций подбирает и предлагает пользователю релевантный контент, основываясь на своих знаниях о пользователе, контенте и взаимодействии пользователя и контента. Основная задача рекомендательной системы состоит в выборе определенных объектов, отвечающих требованиям пользователей. Эти объекты могут быть любыми видами информации или статьями, такими как книги, фильмы, песни, веб-страницы, блоги и т.д.

Формально, задачу нахождения рекомендуемого объекта можно представить следующим образом [2]:

$$\forall\, u \in U,\ s'u = \text{argmax} s \in S\ h(u,s).$$

где U – множество пользователей, S – множество объектов, которые могут быть рекомендованы пользователю, h – функция, определяющая насколько некоторый объект s удовлетворяет некоторого пользователя u. Таким образом, необходимо выбрать такой объект s’ ∈ S, при котором значение удовлетворенности для каждого пользователя u ∈ U максимально.

Существует множество методов построения рекомендательных систем (рис 1.1), но выделяют три основных вида.

1. Фильтрация содержимого (content - based):

   Система учится рекомендовать элементы, похожие на те, которые пользователю нравились в прошлом. Сходство предметов рассчитывается на основе характеристик, присущих каждому элементу. Например, если пользователь положительно оценил фильм, относящийся к жанру комедии, система может научиться рекомендовать другие фильмы этого жанра.

2. Коллаборативная фильтрация (collaborative filtering)

   Самая простая и классическая реализация этого подхода заключается в том, что система делит пользователей на группы по схожим интересам

и затем рекомендует им то, что просматривали/покупали/заказывали другие люди из этого сегмента.

Коллаборативная фильтрация предполагает наличие матрицы оценок User-Item. Такие рекомендации являются результатом «коллаборации» множества пользователей. Отсюда и название метода.

3. Гибридный метод (hybrid recommender)

   Некоторые рекомендательные системы используют комбинации подходов фильтрации содержимого и коллаборативной фильтрации, называемые гибридными методами. Они позволяют в той или иной мере избежать недостатки обоих подходов.

Во всех этих методах используются разные подходы к рекомендациям, и все они имеют свои сильные и слабые стороны, что делает некоторые более подходящими в определенных областях, нежели другие. Рассмотрим эти типы систем более подробно, а также укажем их достоинства и недостатки.

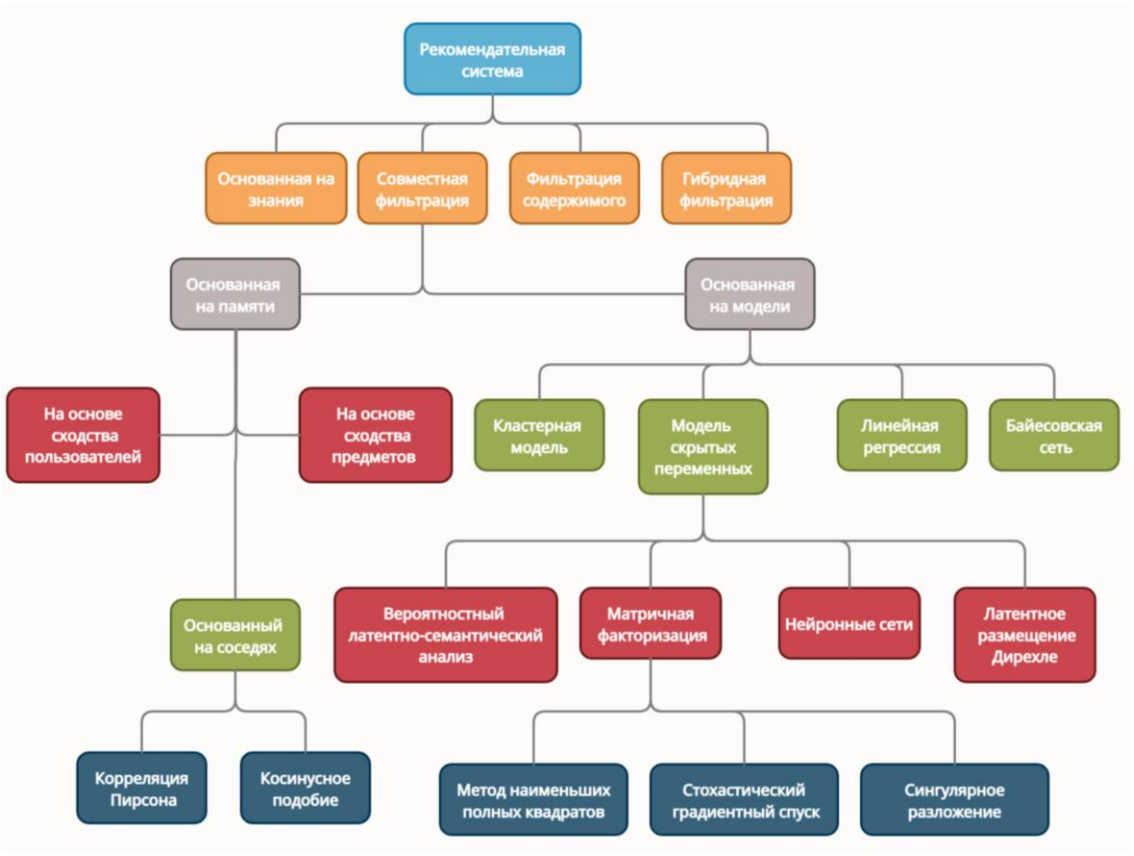


Рис 1.1 Таксономия рекомендательных систем

1.2 Описание различных подходов

1.2.1 Фильтрация на основе контента

Системы, использующие подход, основанный на содержании, использует свойства/параметры объектов, которые мы хотим рекомендовать, чтобы предлагать другие, похожие на те, что нравятся пользователю, на основе его предыдущих действий или явной обратной связи. По мере того как пользователь вводит больше данных или предпринимает действия по этим рекомендациям, механизм становится все более точным.

Процесс рекомендаций, используемый в этом методе, можно условно разделить на три части:

1. **Анализ объектов (того, что будем рекомендовать)**: здесь основная задача состоит в том, чтобы проанализировать содержимое элементов и извлечь из элемента информацию или конкретные характеристики с помощью специальных методов.
2. **Создание профиля**: создаем профиль пользователя после обобщения данных, собранных на основе его предпочтений.
3. **Компоненты фильтрации**: этот процесс будет пытаться сопоставить характеристики профиля пользователя с характеристиками элементов. А затем система порекомендует предметы, которые подходят пользователю.

То есть, эта система работает с данными, предоставленными пользователями явно (рейтинг) или неявно (клик мышкой, длительность просмотра веб-страницы и другие записи, характеризующие поведение пользователя). Из этих данных строится профиль каждого пользователя. Предметы рекомендуются пользователям на основе их сходства: сопоставляя их характеристики. Например, жанр, цвет, производитель. Другими словами, система рекомендует элементы, аналогичные тем, которые нравились пользователю в прошлом, и эта ситуация представлена на рисунке 1.2.

Эффективность фильтрации на основе содержимого зависит от понимания содержимого элементов. Основная цель этого типа - узнать, как различать предметы и как они соотносятся друг с другом. Существует множество методов для извлечения признаков у элементов и последующей обработки: векторизация, TF-IDF, Word2Vec.

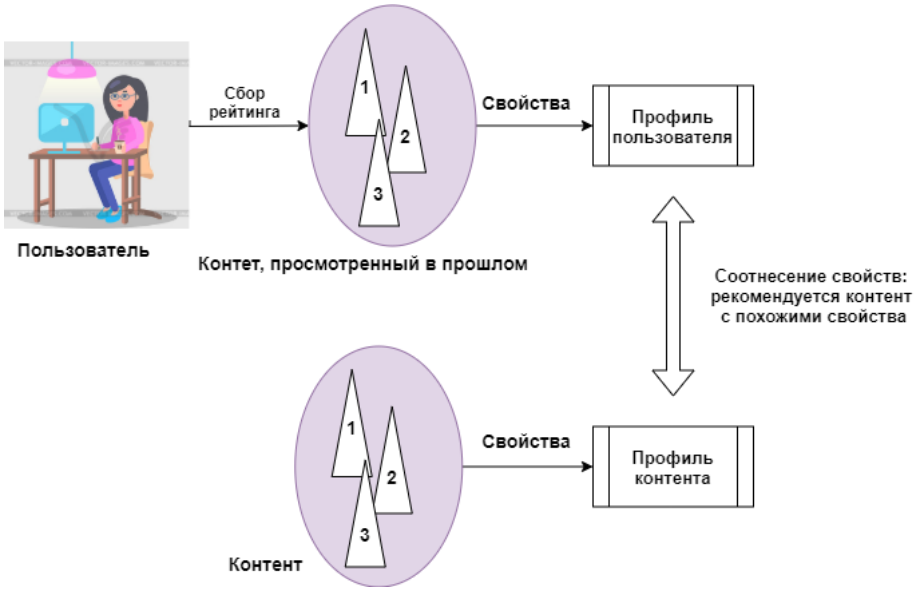


Рис 1.2 Фильтрация на основе содержимого

Принятие парадигмы рекомендаций, основанных на содержании, имеет несколько преимуществ [3][4].

Во-первых, это **независимость пользователя**: модель не требует никаких данных о других пользователях, так как рекомендации специфичны для этого пользователя. Это упрощает масштабирование для большого количества пользователей. Другими словами, рекомендации пользователю не зависят от оценок остальных пользователей

Кроме того, модель может отражать конкретные интересы пользователя и **может рекомендовать нишевые элементы**, которые интересны очень немногим другим пользователям, иначе говоря, такие системы не страдают от проблемы неоцененных объектов.

Но у данной модели есть и существенный недостатки. Поскольку представление характеристик/параметров объектов в некоторой степени разрабатывается вручную, этот метод требует большого **знания предметной**

области. Следовательно, модель может быть настолько хороша, насколько хороши ее характеристики, созданные вручную.

Так же система может давать рекомендации только на основе существующих интересов пользователя. Другими словами, она имеет ограниченные возможности для расширения существующих интересов пользователей т.к. **не рекомендуют что-либо кардинально новое**.

Другим недостатком content-based подхода является необходимость собрать достаточное количество оценок (которое зависит от предметной области разрабатываемой платформы), прежде чем система рекомендаций сможет действительно понять предпочтения пользователя и предоставить точные рекомендации. Поэтому, когда доступно мало оценок, как **для нового пользователя, система не сможет дать надежные рекомендации**.

1.2.2 Коллаборативная фильтрация

Коллаборативная фильтрация (КФ) наиболее популярная и широко распространенная техника в рекомендательных системах. Вместо сопоставления интересов пользователя с объектами, при совместной фильтрации выполняется поиск похожих пользователей или объектов. То есть логика совместной фильтрации заключается в том, что похожим пользователям нравятся одинаковые вещи. Если два пользователя дают одинаковую оценку для одного элемента или ведут себя одинаково, они будут оценивать другие элементы таким же образом. По-другому такой алгоритм называется User-Based (дословно переводится с английского как «на основе пользователя»).

Система КФ включает в себя три основных этапа: начинается со сбора пользовательских оценок по элементам (items) для построения матрицы оценок [8]. Эта матрица используется на втором этапе для поиска соседей посредством вычисления сходства между пользователями или элементами. И, наконец, этап прогнозирования с использованием любых методов агрегирования. На этом этапе выполняется прогнозирование рейтинга для

элементов без оценки, и выбора N-первых элементов в качестве рекомендаций для конкретного пользователя. Интуиция, лежащая в основе этого подхода, заключается в том, что пользователи, которые имели аналогичные предпочтения в прошлом, будут иметь аналогичный интерес в будущем.

Существует два основных подхода к коллаборативной фильтрации

1. Методы, **основанные на модели**: предоставляет рекомендации, измеряя параметры статистических моделей для оценок пользователей, построенных с помощью таких методов как, метод байесовских сетей, кластеризации и другие.
2. Методы, **основанные на памяти**: корреляции между пользователями и элементами используется напрямую, чтобы давать рекомендации пользователю.

Второй подход также подразделяется на два: (рис 1.3) на основе сходства пользователей (**user-based**) и метод на основе сходства объектов (**item-based**).

Рис 1.3. User-Based и Item-Based подходы в коллаборативной фильтрации

Коллаборативная фильтрация имеет ряд преимуществ. Во-первых, **не требуется знание предметной области**, как это было в случае content-based подхода.

Также, модель может помочь пользователям открыть для себя **новые интересы**. По отдельности система машинного обучения может не знать, что пользователь заинтересован в данном элементе, но все же может

рекомендовать его, потому что аналогичные пользователи заинтересованы в нем.

Одной из самых острых проблем является «**проблема холодного старта**». Это означает, что пользователю необходимо сначала оценить достаточно большое количество объектов, прежде чем рекомендательная система сможет дать хорошие рекомендации.

Однако следующие методы могут в некоторой степени решить проблему холодного запуска: активное обучение, LDA, обогащение профилей элементов дополнительным источником данных.

Другой важной проблемой является **разреженность данных**, что очень сильно влияет на производительность таких рекомендательных систем, так как это сильно снижает вероятность нахождения пользователей (или объектов) со схожими оценками. Эта проблема возникает вследствие того, что рекомендательные системы обычно работают с огромным количеством объектов и пользователей. Например, конкретный пользователь просто физически не может посмотреть и оценить большинство выходящих в прокат кинофильмов.

1.2.3 Гибридный метод

Существуют разные достоинства и недостатки как для совместной фильтрации, так и для фильтрации на основе содержимого. Поэтому в некоторых случаях, когда доступны данные, у нас есть возможность использовать разные типы рекомендательных систем для одной и той же задачи. В таком случае существует много возможностей для комбинирования различных типов систем чтобы достичь лучшей производительности за счет исключения недостатков каждой техники в отдельности.

Берк в своей книге [7] разделил гибридные методы на семь различных типов:

1. Смешанный: рекомендации нескольких разных систем отображаются в одно и то же время

2. Взвешенный: оценки (или голоса) нескольких рекомендательных методов объединяются в одну рекомендацию.
3. Комбинация признаков: характеристики из разных источников данных рекомендаций объединены в единый алгоритм.
4. Каскад: одна рекомендательная система уточняет данные, данные другой.
5. Аугментация признаков: выходные данные одного метода используются как входные данные для другого.
6. Мета-уровень: модель, изученная одним рекомендателем, используется в качестве исходных данных для другого.
7. Переключение: Система переключается между техниками рекомендаций в зависимости от текущей ситуации.

Берк и его коллеги попытались сравнить эффективность и производительность этих различных типов гибридных рекомендательных систем. По результатам экспериментов гибриды показали преобладание над базовыми системами рекомендаций. Лучшими были признаны аугментации и каскадирование. Это доказало, что основным достоинством гибридного метода является возможность комбинировать преимущества других методик, при этом исключая их недостатки. Но многие компоненты таких систем обладают относительной точностью и согласованностью, и их характеристики следует тщательно учитывать, чтобы создать эффективный алгоритм. Именно поэтому сложность разработки можно назвать главным минусом гибридных систем.

1.3 Алгоритмы и метрики

Рассмотрим методы, которые используются в вышеупомянутых подходах.

1.3.1 Алгоритмы рекомендательных систем, основанных на содержании

Хорошо известной техникой, используемой при реализации рекомендательной системы, основанной на содержании, являются

векторизация **TF-IDF**. TF-IDF (term frequency-inverse document frequency) предназначен для выделения ключевых слов текста. Это простой и удобный способ оценить важность термина для какого-либо документа относительно всех остальных документов. Тут учитывается частота встречаемости слова в нескольких текстах. Слово имеет большое значение, если он часто встречается в одном тексте и редко в других. TF (term frequency) вычисляет частоту в тексте (1.3.1), а IDF (inverse document frequency) показывает частоту документов (1.3.2). Таки образом можно представить в виде формулы:

$$TF(x,A) = \frac{fr(x,A)}{max_{y \in A} fr(y,A)},$$

$$IDF(x) = \frac{N}{n(x)},$$

где fr(x,A) – количество слов x в документе A; N- количество документов в наборе, n(x) – количество документов, в которых встречается слово x.

Вычисление TF может принимать значения только от 0 до 1. Сам коэффициент TF - IDF вычисляется как умножение TF на IDF. Для сопоставления двух текстов, их можно представить в виде векторов в многомерном пространстве.

Векторы TF-IDF обычно большие (в корпусе: под «корпусом» понимают «унифицированный, структурированный и размеченный массив языковых данных в электронном виде, встречаются тысячи слов) и очень разреженные (в каждом документе используется только часть слов). В практических приложениях такие длинные и разреженные векторы не только вызывают проблемы в отношении требований к производительности и памяти, но также приводят к так называемому эффекту переобучения. Чтобы сделать их более компактными и удалить из вектора нерелевантную информацию, можно применить дополнительные методы.

Стоп-слова и стемминг. Самый простой способ - удалить так называемые стоп-слова. В русском языке это, например, предлоги и союзы, такие как «и», «в» или «потому что», которые можно удалить из векторов

документов, поскольку они будут присутствовать почти во всех документах. Другой широко используемый метод называется стемминг, который направлен на замену вариантов одного и того же слова их общей основой (корневым словом). Слово «основание», например, было бы заменено на «основа», «пошел» на «идти» и так далее. Эти методы дополнительно уменьшают размер вектора и в то же время помогают улучшить процесс сопоставления в тех случаях, когда основы слов также используются в профиле пользователя.

Фразы. Дальнейшее возможное улучшение точности представления состоит в использовании «фраз как терминов», которые более описательны для текста, чем отдельные слова. Обнаружение фраз может быть выполнено путем поиска списков, составленных вручную, или путем применения методов статистического анализа.

Когда проблема выбора элементов при совместной фильтрации может быть описана как «рекомендовать элементы, которые понравились схожим пользователям», рекомендация, основанная на содержании, обычно описывается как «рекомендовать элементы, подобные тем, которые нравились пользователю в прошлом». Для этой задачи также, как и для совместной фильтрации, может использоваться метод поиска k ближайших соседей, который подробно описан в следующем пункте. (1.3.2)

1.3.2 Алгоритмы рекомендательных систем, основанных на совместной фильтрации

В коллаборативной фильтрации используются такие техники, как обучение ассоциативным правилам, классификация, кластеризация, статистика и алгоритмы машинного обучения.

Классификация на основе ближайших соседей

K-ближайший сосед (kNN) - один из самых известных и базовых алгоритмов, используемых в машинном обучении и интеллектуальном анализе данных. Алгоритм kNN представляет собой простой, но эффективный

метод непараметрической классификации, основанный на оценивании сходства объектов [9].

Учитывая точку, которую необходимо классифицировать, классификатор kNN находит k ближайших точек (ближайших соседей) из обучающих записей. Затем он присваивает метку класса в соответствии с метками классов своих ближайших соседей. Основная идея заключается в том, что, если запись попадает в определенное окружение, где преобладает метка класса, это происходит потому, что запись, вероятно, принадлежит к тому же самому классу. Возможно самая сложная проблема в kNN - это выбрать значение k. Если k слишком мало, классификатор будет чувствителен к точкам шума. Но если k слишком велико, соседство может включать слишком много точек из других классов. Поскольку kNN не строит модели явно, он считается “ленивым учеником”. В отличие от “активных учеников”, таких как деревья решений или системы на основе правил, классификаторы kNN оставляют многие решения на этапе классификации. Следовательно, классификация неизвестных записей относительно дорога. Ближайшее соседство - один из наиболее распространенных подходов к классификации и, следовательно, к проектированию рекомендательных систем. Подход kNN, хотя и простой и интуитивно понятный, показывает хорошие результаты по точности и хорошо поддается усовершенствованию.

Метод k-средних

Метод k-средних (англ. k-means) — один из простейших и наиболее популярных методов кластеризации или кластерного анализа, в котором число кластеров k выбирается заранее. Далее задача состоит в том, чтобы сегментировать входящие значения на подмножества S_1, ..., S_k . таким образом, чтобы минимизировать полную сумму квадратов расстояний от каждой точки до среднего значения назначенного ей кластера.

Назначить n точек данных k группам можно самыми разными способами, вследствие чего найти оптимальную конфигурацию групп очень

трудно. Рассмотрит итеративный алгоритм, который обычно находит хорошую конфигурацию [10].

1. Начать с множества k средних, т.е. точек в d-мерном пространстве.
2. Назначить каждую точку ближайшему среднему значению.
3. Если ни у одной точки ее значение не изменилось, то остановиться и сохранить группы.
4. Если назначение одной из точек изменилось, то пересчитать средние значения и вернуться к шагу 2.

Однако у этого алгоритма есть несколько недостатков: он предполагает предварительное знание данных для выбора подходящего k; конечные кластеры очень чувствительны к выбору начальных центроидов; и он может создавать пустой кластер.

Матричная факторизация

Конкурс Netflix Prize, завершившийся в 2009 году, показал, что передовые методы матричной факторизации, которые использовались многими участвующими командами, могут быть особенно полезны для повышения точности прогнозирования рекомендательных систем. Методы матричной факторизации позволяют встраивать данные больших размеров в пространства меньших размеров, которые смягчают эффекты из-за шума, выявляют скрытые отношения или облегчают дальнейшую обработку.

Сингулярное разложение (SVD) хорошо зарекомендовало себя для выявления скрытых семантических факторов [16]. В области кино такие автоматически определяемые факторы могут соответствовать очевидным аспектам фильма, таким как жанр или тип (драма или действие), но они также могут быть не интерпретируемыми. Применение SVD в области совместной фильтрации требует факторизации матрицы рейтинга пользовательских элементов.

Каждому элементу i соответствует вектор $q_i \in R^f$, а каждому пользователю u соответствует вектор $p_u \in R^f$. Для данного элемента i элементы

q_i измеряют на сколько элемент обладает определенными факторами, положительными или отрицательными. Для данного пользователя u элементы p_u измеряют степень интереса пользователя к элементам. Результирующее скалярное произведение $q_i{}^T p_u$ отражает взаимодействие между пользователем u и элементом i - общий интерес пользователя к характеристикам элемента обозначается r_{ui}. Таким образом формула представлена в таком виде:

$$r_{ui} = qi^T p_u$$

Рекомендательная система, используя это уравнение, может легко оценить рейтинг, который пользователь поставит любому элементу.

Таким образом, метод матричного разложения SVD в контексте рекомендательных систем ищет новые факторы и помогает преодолеть проблему размерности, преобразовывая исходное многомерное пространство в более низкое

1.3.3 Метрики

Наиболее популярные метрики, которые используются для вычисления сходства между векторами: евклидово расстояние, косинусная мера, корреляция Пирсона и другие.

Самый простой и наиболее распространенный пример меры расстояния - это **евклидово расстояние**:

$$d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

где n - количество измерений (атрибутов), а x_k и y_k - k-е атрибуты (компоненты) объектов данных x и y, соответственно.

Другой очень распространенный подход - рассматривать элементы как векторы документов в n-мерном пространстве и вычислять их подобие как косинус угла, который они образуют:

$$cos(x,y) = \frac{(x \bullet y)}{||x||||y||}$$

где • обозначает скалярное произведение векторов, ‖x‖ - норма вектора x. Это сходство известно, как **косинусное подобие** или метрика L2.

Сходство между элементами также может быть задано их корреляцией, которая измеряет линейную взаимосвязь между объектами. Хотя есть несколько коэффициентов корреляции, которые могут применяться, **корреляция Пирсона** является наиболее часто используемой. Учитывая ковариацию Σ точек данных x и y и их стандартное отклонение σ, мы вычисляем корреляцию Пирсона, используя:

$$Pearson(x,y) = \frac{\Sigma(x,y)}{\sigma_x \times \sigma_y}$$

Глава 2. Практическая реализация

В рамках данной работы во время изучения основ теоретической части сначала была реализована рекомендательная система, которая, основываясь на данных тестирования студентов, может использоваться в образовательном контенте для повышения качества обучения (пункт 2.1). Данный этап изучения предметной области и исследования пришелся на первый семестр обучения. В дальнейшем во втором семестре с увеличением полученных навыков, изучением алгоритмов машинного обучения и углублением в тему разработки рекомендательных систем был разработан алгоритм рекомендаций разбиения онлайн-курсов на категории на основе их описания (пункт 2.2).

Рекомендательные системы могут применяться в различных ситуациях. Основные из них:

1. Рекомендация для конкретного пользователя на основе правильности его ответов на вопросы со списком тем для последующего изучения или повторения
2. Рекомендация для учителя, которая показывает учащихся с максимально похожим уровнем знаний. Может применяться для разбивки учеников на группы, а также для дальнейшего анализа и выявления факторов и закономерностей, влияющих на их уровень. К примеру, может оказаться, что у группы учеников с низкой успеваемостью одинаковый график учебы или один преподаватель.
3. Рекомендация вопросов, на которые ученик вероятней всего ответит или не ответит, для формирования дальнейшего тестирования.
4. Рекомендации категорий курсов для веб-платформ, нацеленных на образовательный контент.
5. В качестве логической основы для других рекомендаций.

2.1 Рекомендательная система на основе данных о тестировании

Данная система рекомендует студентам темы для повторения, а также возможный вариант разбивки учеников на группы. Создавалась в процессе изучения основ теоретической части для изучения методов работы с большими данными, их группировки и применения базового алгоритма k-ближайших соседей. Система может быть прикреплена к приложениям/веб-платформам на которых проводится онлайн-тестирование для того, чтобы студенту и преподавателю автоматически формировались рекомендации по окончанию контрольной.

2.1.1 Данные

Для практической реализации метода рекомендаций, было принято решение воспользоваться свободно распространяемыми данными (https://www.kaggle.com/johnweek/students-test-results) результатов тестирования студентов по различным темам.

В выбранных данных представлено 10000 оценок по 100 вопросам от 1000 студентов. Причем каждый пользователь в этих данных ответил не менее, чем на 20 вопросов.

Данные представлены в виде dataframe, где индексы - это имена студентов, названия столбцов - id вопросов. Структура данных содержит информацию о правильности 1, неправильности 0 или отсутствия ответа Nan на вопрос студентом. Пропущенное значение Nan означает, что данный вопрос еще не присутствовал при тестировании студента.

Также нам известны, какие области знаний затрагивает каждый вопрос. Эта информация представлена в виде словаря:
{id_question: [Theme_1, Theme2, Theme3, ...]}, где id_question - это номер вопроса, Theme_x - название темы.

Выборка данных показана на рисунке (2.1):

	question 0	question 1	question 2	question 3	question 4	question 5
Paul Jones	1.0	NaN	NaN	1.0	1.0	0.0
James Johnson	NaN	NaN	0.0	0.0	1.0	1.0
Nicholas Peck	0.0	0.0	0.0	0.0	NaN	0.0
Gabriel Hall	0.0	NaN	NaN	NaN	1.0	0.0
Nicolas Castillo	1.0	0.0	1.0	0.0	1.0	1.0

Рис 2.1. Выборка данных

2.1.2 Подход к рекомендациям

В процессе данного исследования сначала был разработан простой механизм автоматического анализа результатов тестирования, который на основании словаря с номерами вопросов и соответствующих тем, и на основании ответов студентов на вопросы формирует для пользователей рекомендации.

Рекомендации представляют собой:

1) список тем, рекомендованных для повторения
2) список вопрос, на которые студент вероятнее всего ответит/не ответит
3) список учеников, имеющих приблизительно одинаковый уровень знаний

Алгоритм работы программы

На вход программа получает фрейм с результатами тестирования и словарь с темами, которые затрагивает каждый вопрос, а также вес для каждого вопроса. Данные прошли небольшую предварительную обработку. Для обучения модели были отброшены те строки, где количество неизвестных значений превышало 80% от общего числа вопросов. Была написана функция, которая создает словарь для каждого ученика, где ключами выступают темы вопросов, а значениями список всех набранных баллов по каждому вопросу, который затрагивает эту область знаний. Простой пример, студент ответил правильно на 2 вопроса, а на 3 ответ был неверным. При этом первый вопрос

затрагивал такие темы, как “Алгоритмы, Программирование, Графы”, второй “Алгоритмы, Статистика”, а третий “Алгоритмы, Графы”. Для этого студента формируется словарь {Алгоритмы: [1, 1, 0], Статистика: [1], Графы: [1,0]}.

Далее с учетов веса каждого балла для каждой темы будет вычислена средневзвешенная оценка по формуле:

$$O_{cp} = (O_1*P_1 + O_2*P_2 + \ldots + O_N*P_{N)} / (P_1 + P_2 + \ldots + P_N),$$

где O_N – это оценки за задания, P_N – вес оценок.

```
dicts = {}
for name in names:
    dict_name = []
    for i in range(0, len(list(themes.values()))):
        for j in range(0, len(list(themes.values())[i])):
            dict_name.append({list(themes.values())[i][j] : df.loc[name][i]})

    dict_name = [i for i in dict_name]
    grouped_data = defaultdict(list)
    for d in dict_name:
        for k, v in d.items():
            if v != None:
                grouped_data[k].append(v)

    mean_data = {k: wavg([int(i) for i in v]) for k, v in grouped_data.items() }
    dicts[name] = mean_data
```

Рис 2.2. Формирование словаря для каждого ученика со средневзвешенной оценкой по каждому предмету

Темы, которые получили наименьшую оценку (порог устанавливается на усмотрение преподавателя), будут предложены студенту. Рекомендации выглядят таким образом:

```
Sarah West
You need read more about Program Analysis
You need read more about Algorithms
```

Рис 2.3. Пример рекомендаций

Матрица с результатами тестирования идеально подходит, чтобы применить алгоритм k ближайших соседей. Поэтому для рекомендаций разбиения учеников на группы, как и для списка вопросов использовался именно этот алгоритм из библиотеки sklearn.

```
neigh = neighbors.NearestNeighbors(n_neighbors=5)
neigh.fit(people)
neighs = neigh.kneighbors(men, return_distance=False)
```

Рис 2.4. Нахождение k-ближайших соседей

Пример рекомендаций:

```
These people are similar. They have the same level of knowlegde
Ryan Nash MD
Michael Hunter
Christopher Weber
Andre Wall
Matthew Elliott
Gerald Tucker
```

Рис 2.5. Пример рекомендаций

2.2 Рекомендательная система онлайн-курсов

В дальнейшем с погружением в изучение алгоритмов и методик рекомендательных систем, была разработана рекомендательная система, которая предлагает онлайн-курсы, основываясь на идентификаторе курса, который ввел пользователь. Система может использоваться на платформах с онлайн-курсами: когда пользовать заходит на определённый курс, ему рекомендуются похожие курсы. Хотелось бы подробнее остановиться именно на этой реализации потому, что она более полно отражает полученные навыки и методику построения рекомендательных систем. Прикладываю ссылку на репозиторий: https://github.com/vladanills/vkr_recommender-system

2.2.1 Данные

Для практической реализации метода рекомендаций, было принято решение воспользоваться свободно распространяемыми данными, представляющими собой список и описание курсов. Для получения данных курсов с веб-платформы был запущен запрос RestApi, который выглядит следующим образом:

```
http://api.pluralsight.com/api-v0.9/courses
```

Рис 2.6. Запрос

Запрос возвращает файл Coursers.csv, который содержит информацию о названии 8000 курсов и их описаний. Структура файла содержит:

1. CourseId - идентификатор курса,
2. CourseTitle - название курса,
3. DurationInSeconds - продолжительность,

4. ReleaseDate - дата выхода,
5. Description - описание,
6. AssessmentStatus - оценка,
7. IsCourseRetired - статус доступности.

При первичном анализе полученных данных можно заметить, что текстовое описание содержится только в столбцах "CourseId", "CourseTitle" и "Description". Именно эти столбцы представляют интерес при построении нашей системы рекомендаций. С текстовыми данными из этих столбцов можно построить векторы слов, которые будут использоваться нашей моделью при прогнозировании результатов. Кроме того, большая часть информации представлена только в столбце «Description». Следовательно, курсы без описания будут исключены из обучения. Также не стоит рекомендовать старые курсы, которые недоступны в данный момент. Это легко узнать благодаря столбцу “IsCourseRetired”, который показывает доступен ли курс в настоящее время на веб-сайте или нет. Более того потребуется предварительная обработка данных, как как в них присутствуют некоторые лишние элементы (например, “-”) и стоп-слова.

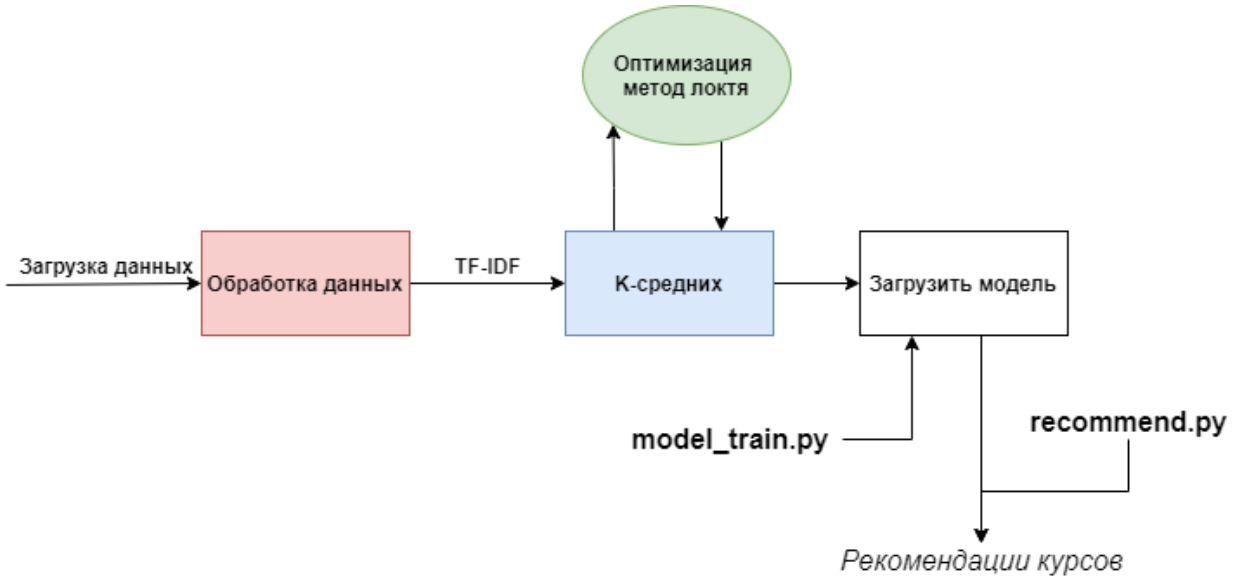
Выборка данных показана на рисунке (2.2):

	CourseId	CourseTitle	DurationInSeconds	ReleaseDate	Description	AssessmentStatus	IsCourseRetired
0	abts-advanced-topics	BizTalk 2006 Business Process Management	22198	2008-10-25	This course covers Business Process Management...	Live	no
1	abts-fundamentals	BizTalk 2006 Fundamentals	24305	2008-06-01	Despite the trend towards service-oriented arc...	Live	no
2	agile-team-practice-fundamentals	Agile Team Practices with Scrum	13504	2010-04-15	This course is much different than most of the...	Live	no
3	appsrv-fundamentals	Windows Server AppFabric Fundamentals	0	2000-01-01	NaN	None	yes
4	aspdotnet-advanced-topics	ASP.NET 3.5 Advanced Topics	21611	2008-12-05	This course covers more advanced topics in ASP...	Live	no

Рис 2.7. Выборка данных

2.1.2 Подход к рекомендациям

Архитектура рекомендательной системы показана на следующей схеме:



Сначала данные были предварительно обработаны, после чего векторизованы с помощью алгоритма TF-IDF. Затем для формирования кластеров применялся алгоритм k-средних, в котором для нахождения нужного k была применена оптимизация методом локтя (elbow method). После чего сохраненная модель была использована для выдачи рекомендаций.

Алгоритм работы программы

На вход программа получает scv файл. В первую очередь нужно произвести небольшую предварительную обработку данных. Сначала были удалены все строки с пустыми (Nan) значениями, потому что они не несут никакой ценности для дальнейшего формирования рекомендаций. После чего из столба с описанием были удалены все местоимения и апострофы, а из идентификаторов курса были удалены все “-”, так как они присутствуют в каждой строке. Соответствующим образом объединим столбцы, содержащие описание, идентификатор курса и заголовок и удалим все символы, кроме букв и цифр. После выполнения этих шагов по очистке вышеуказанных данных фрейм содержит все необходимые описания слов, относящиеся к курсу. Полученный фрейм выглядит подобным образом:

```
0       abts advanced topics BizTalk 2006 Business Pro...
1       abts fundamentals BizTalk 2006 Fundamentals De...
2       agile team practice fundamentals Agile Team Pr...
4       aspdotnet advanced topics ASPNET 35 Advanced T...
5       aspdotnet ajax advanced topics ASPNET Ajax Adv...
                              ...
8011    nunit moq mocking Mocking with Moq and NUnit W...
8012    identity access management aws users Identity ...
8013    secure coding using components known vulnerabi...
8014    aws s3 implementing Implementing Amazon S3 Sto...
8015    identity access management aws roles groups Id...
```

Рис 2.8. Промежуточные данные

Можно перейти к векторизации этого текста и дальнейшему обучении модели.

Для осмысленного представления данных было выбрано использование весов **tf-idf**, которые означают важность термина в документе. Это статистическая мера важности слова в документе. Данное представление было подробно описано ранее. Повторюсь, что Tf измеряет частоту употребления терминов в документе. И idf измеряет важность данного термина в данном корпусе. Для данного преобразования текстовых данных в вектора в Python существует библиотека sklearn, а конкретно класс TfidfVectorizer, который может токенизировать документы, изучать словарный запас и обратные весовые коэффициенты частоты документов.

После этого теперь возможно передать эти данные прямо в алгоритм обучения **k-средних** с помощью класса Kmeans из библиотеки sklearn. Но встает вопрос о выборе значения k для данного алгоритма. На веб-платформе курсы разделены на 8 категорий, поэтому можно предположить, что k=8. Прогоним алгоритм k-средних для данного k и сопоставим каждому кластеру полученный термин.

```
model = KMeans(n_clusters=true_k, init='k-means++', max_iter=500, n_init=15)
model.fit(X)

print("Категории для каждого кластера:")
order_centroids = model.cluster_centers_.argsort()[:, ::-1]
terms = vectorizer.get_feature_names()
for i in range(true_k):
    print("Cluster %d:" % i),
    for ind in order_centroids[i, :15]:
        print(' %s' % terms[ind]),
    print
```

Рис 2.9. k-means

После проверки возможности прогнозирования нашей модели, обученной соответствующим образом, видно, что все сформированные кластеры не подходят, и некоторые категории курсов повторяются в нескольких кластерах, как видно на рисунке (рис 2.10).

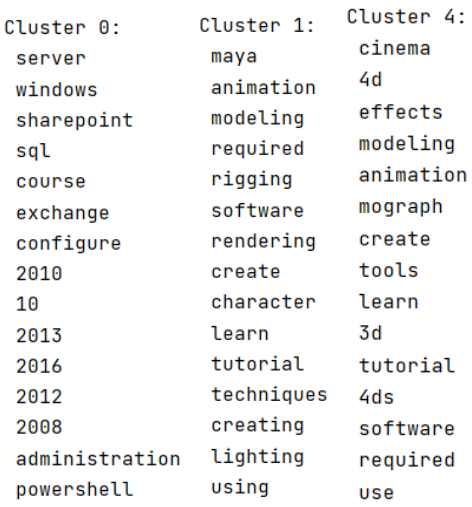


Рис 2.10. Пример сформированных кластеров для k=8

Можно заметить, что категория «творчество-профессионал» разбилась на подкатегории «графический дизайн», «Дизайн фильмов» и «Анимация». Это произошло из-за неравномерности распределения курсов по категориям. В таких категориях курсов, как «бизнес-профессионал» находиться очень небольшое количество курсов, поэтому все термины, связанные с бизнесом, которые не используются часто, могут легко потерять свой вес tf-idf в обучение модели машинного обучения. Очевидно, что изначальное предположение о 8 кластерах неверно и кластеры, полученные на основе этого подхода, могут быть улучшены путем дальнейшего разделения на другие кластеры для получения этих более мелких категорий курсов с меньшим количеством курсов

Существует целый ряд способов выбора числа k. Один из самых простых для понимания предполагает построение диаграммы суммарного квадратичного отклонения, так называемую ошибку (между каждой точкой и средним своей группы), как функции от k, и нахождения “излома” кривой [17]. То есть можно сформировать задачу нахождения k, как задачу оптимизации с минимизацией ошибок. По другому такой метод называется, как метод локтя

(elbow method). После предварительной обработки и векторизации, запустим алгоритм k-средних для каждого потенциально подходящего k, для которого будем сохранять сумму квадратов расстояний.

```
sse = {}

for k in range(1, 31):
    kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=100).fit(X)
    comb_frame["clusters"] = kmeans.labels_
    sse[k] = kmeans.inertia_
```

Рис 2.11. elbow method

После визуализации полученного результата видим следующий график (рис 2.12), на основе которого можно заметить, что относительно лучшие кластеры, примеры которых можно увидеть на рисунке (рис 2.13), получаются при k = 30.

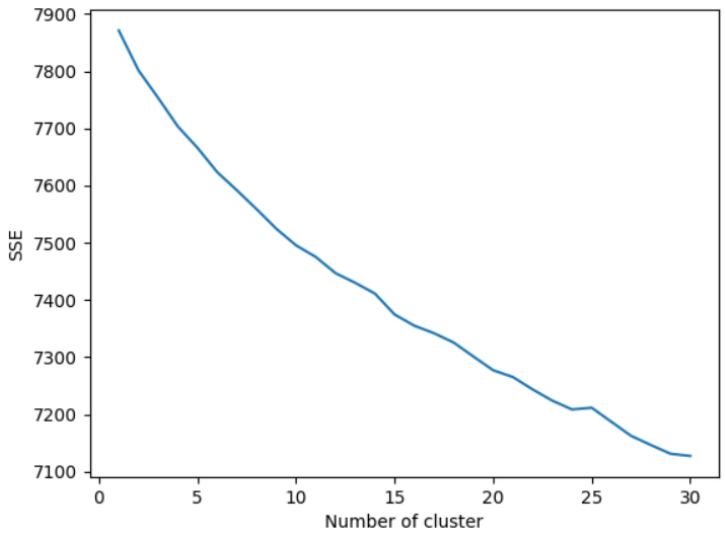


Рис 2.12. График суммы квадратов ошибок

Cluster 19:	Cluster 20:	Cluster 12:
unity	security	data
game	awareness	course
create	information	database
games	cyber	learn
character	attacks	big
fundamentals	network	using
creating	course	analysis
development	threats	access
learn	secure	spark
2d	cissp	use
substance	protect	visualization
player	learn	started
course	risks	hadoop
required	controls	visualizations
software	risk	excel

Рис 2.13. Пример сформированных кластеров для k=30

Теперь перейдем непосредственно к рекомендациям. В первую очередь была создана функция, которая будет предсказывать кластер любого вводимого в него описания. Предпочтительным вводом является ввод, подобный «описанию» из выборки данных.

```
def cluster_predict(str_input):
    Y = vectorizer.transform(list(str_input))
    prediction = model.predict(Y)
    return prediction
```

Рис 2.14. функция

После этого создается новый столбец фрейма данных, а именно «ClusterPrediction», который будет содержать категории для каждого курса на основе их вектора описания. Как оговаривалось ранее нас не интересуют курсы, который уже недоступны для прохождения, поэтому перед присвоением категорий следует отбросить курсы, у которых IsCourseRetired == 'no'. Потом для заполнения столбца с категориями была запущена ранее созданная функция, прогнозирующая категории для каждого курса из фрейма данных. Эти сохраненные категории будут в будущем сопоставляться с входным запросом и его прогнозируемой категорией для выработки рекомендаций.

Последним шагом была создана функция, которая предсказывает категорию курса введенного запроса, имеющего идентификатор курса, и будет рекомендовать несколько случайных курсов из преобразованного ранее фрейма данных, который имеет прогнозируемую категорию для каждого курса.

```
def recommend(str_input):
    temp_df = course_df.loc[course_df['CourseId'] == str_input]
    temp_df['InputString'] = temp_df.CourseId.str.cat(" " + temp_df.CourseTitle.str.cat(" " + temp_df['Description']))
    str_input = list(temp_df['InputString'])

    prediction_inp = cluster_predict(str_input)
    prediction_inp = int(prediction_inp)

    temp_df = course_df.loc[course_df['ClusterPrediction'] == prediction_inp]
    temp_df = temp_df.sample(10)

    return list(temp_df['CourseId'])
```

Рис 2.15. рекомендация курсов

Вывод рекомендаций выглядит следующим образом:

```
For  wp7-core - ['asynchronous-javascript-reasoning', 'windows8-design-to-delivery', 'useful-jquery-plugins', 'fsharp-test-driv
en-development', 'leadership-getting-started', 'building-memory-game-signalr', 'identifying-personally-identifiable-informatio
n', 'tensorflow-building-regression-models', 'windows-embedded81-industry-jumpstart', 'powershell-getting-started']

For  ef41-data-access - ['cplusplus-data-structures-algorithms', 'apache-kafka-getting-started', 'apache-spark-sql-fast-data-ha
ndling-streaming', 'rapidminer-getting-started', 'elasticsearch-analyzing-data', 'dynamic-data-tutorial', 'introduction-motion-
capture-motionbuilder-100', 'hdinsight-deep-dive-storm-hbase-hive', 'oracle-big-data-getting-started', 'd3-getting-started']

For  nosql-big-pic - ['premiere-pro-media-encoder-mastering-compression-settings-2502', 'xaml-patterns', 'cryptography-fundamen
tals-java-dotnet-developers', 'react-native-getting-started', 'presentation-design-introduction', 'incopy-cc-fundamentals', 'me
f', 'inversion-of-control', 'communications-better-technology-deployments', 'oracle-triggers']

For  procedural-ice-modeling-softimage-153 - ['beginners-guide-rotoscopes-softimage-258', 'getting-started-replicators-modo-19
9', 'working-arbor-press-solidworks-1626', 'daydreamer-production-pipeline-6-dynamics-2247', 'custom-shaders-houdini-2282', 'yo
ur-first-day-blender-1786', 'replicating-an-italian-renaissance-era-temple-rhino-1849', 'modeling-options-sketching-features-as
semblies-solidworks-1722', 'windmill-blade-solidworks-1990', 'houdini-designing-vex-driven-digital-assets']

For  beginners-guide-shading-networks-softimage-510 - ['using-ice-create-holographic-display-softimage-1461', 'mental-ray-produ
ction-shaders-softimage-147', 'solidworks-simulation-thermal-analysis', 'modeling-options-sketching-features-assemblies-solidwo
rks-1722', 'solidworks-working-curves-splines', 'solidworks-essentials-basic-part-modeling', 'quick-start-rigging-modo-1-1780',
'crowds-houdini-15-2387', 'character-rigging-softimage-1025', 'solidworks-design-communication-documents-consumer-products']
```

Рис 2.16. Пример рекомендаций

Модель k-средних эффективна, когда дело доходит до прогнозирования категорий курсов, которые находятся в хорошей пропорции и имеют разумное количество связанных с ними ключевых слов. Рекомендации или прогнозы коррелированных курсов, связанных с кластером, не очень хороши. Для идентификаторов, которые относятся к одному кластеру Machine Learning сформировались рекомендации курсов, описания которых не связаны между собой. Это хорошо видно на примере ниже:

```
For  play-by-play-machine-learning-exposed - ['play-by-play-getting-started-project-online-dux-raymond-sy', 'play-by-play-websi
te-security-review-troy-hunt-lars-klint', 'play-by-play-salesforce-integration-on-a-budget', 'play-by-play-understanding-apex-e
nterprise-patterns-separation-concerns-salesforce', 'play-by-play-rob-sullivan', 'play-by-play-discovering-powershell-minasi',
'play-by-play-jitjea-snover', 'play-by-play-implementing-sustainable-scalable-salesforce-governance', 'play-2-java', 'play-by-p
lay-salesforce-reusable-lightning-components-builder']

For  microsoft-cognitive-services-machine-learning - ['tekpub-ravendb', 'identifying-fixing-performance-issues-caused-parameter
-sniffing', 'your-first-day-pftrack-1762', 'serialization-dotnet-4-5', 'electronics-fundamentals', 'wcf-design-concepts', 'adob
e-target-using-analytics-shared-audiences', 'sculpting-human-noses-mudbox-1177', 'toxik-node-reference-library-channel-stream-n
odes-164', 'indie-game-dev-pipeline-1-visual-development-1531']

For  python-scikit-learn-building-machine-learning-models - ['autodesk-alias-sketching-sports-car', 'using-morph-magic-daemons-
realflow-413', 'premiere-elements-fundamentals', 'ruby-on-rails3-fundamentals', 'material-creation-workflows-udk-1069', 'csharp
-extension-methods', 'javascript-asynchronous-modern', 'prezi-getting-started', 'introduction-mel-61', 'emc-xtremio-implementat
ion']

For  pandas-data-wrangling-machine-learning-engineers - ['apache-kafka-getting-started', 'plotly-building-data-visualizations',
'mapreduce-applying-common-data-problems', 'brightstardb-introduction', 'tableau-desktop-building-effective-data-communication
s', 'data-you-can-see', 'introduction-motion-capture-motionbuilder-100', 'seaborn-visualizing-statistical-data', 'apache-flink-
stream-processing-getting-started', 'getting-started-power-bi-business-professionals']

For  xgboost-python-scikit-learn-machine-learning - ['custom-templates-archicad-2161', 'riak-introduction', 'flux-redux-masteri
ng', 'implementing-custom-middleware-components-aspdotnet-core', 'iac-wcf', 'marvelous-designer-creating-victorian-style-gown',
'adobe-cc-designing-shrink-sleeves', 'asynchronous-messaging-rabbitmq-easynetq', 'creating-configuring-new-websites-iis', 'draw
ing-human-head-237']
```

Рис 2.17. Пример рекомендаций

Это подчеркивает тот факт, что даже после минимизации потерь сформированный кластер не так точен. С жестким заданием шага, подобный подход для формирования кластеров, является довольно грубым, но дает хорошее представление о реализации рекомендательных систем с алгоритмами кластеризации. По сути, для будущих улучшений механизм присвоения категорий и модель, используемые для обучения, могут быть изменены. Могут быть приняты более продвинутые и сложные механизмы

тематического моделирования, такие как скрытое распределение Дирихле (LDA).

Заключение

В данной работе была представлена рекомендательная система (РС), которая основываясь на введенном идентификаторе, определяет тематику запроса и рекомендует существующие онлайн-курсы. Также была разработана РС, которая опираясь на результаты тестирования предлагает для студентов темы для повторения, а для учителей возможный вариант разбиения на группы по уровню знаний.

В процессе проведения исследования был произведен обзор предметной области рекомендательных систем, были изучены основные методы в разработке РС, главные подходы к их построению, а также модули и библиотеки Python для работы с большими массивами данных (numpy и pandas). Кроме того, были рассмотрены популярные алгоритмы машинного обучения, которые активно применяются при реализации различных систем. Также были проанализированы существующие стратегии решения задач рекомендаций в образовательной среде.

Проделанная мной работа является начальной точкой в дальнейшем изучении и реализации рекомендательных систем в образовании. Реализация РС продемонстрировала работу популярных алгоритмов, логику подбора параметров, а также выявила проблемы, которые могут возникнуть при применении различных подходов. Детальное изучение существующих методов позволит произвести усовершенствование существующих решений.

Список Литературы

[1] C. Anderson, The long tail: Why the future of business is selling less of more. Hachette Books, 2006.

[2] Adomavicius G. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions / G. Adomavicius, A. Tuzhilin // IEEE Transactions on Knowledge and Data Engineering. – 2005. – Vol. 17, №6. – P. 734-749

[3] Francesco Ricci , Lior Rokach, Bracha Shapira, Paul B. Kantor. Recommender Systems Handbook

[4] Машинное обучение. Рекомендательные системы [Электронный ресурс]: Электронный журнал. URL: https://developers.google.com/machine-learning/recommendation

[5] Herlocker, J., Konstan, J., Riedl, J.: Explaining collaborative filtering recommendations. In: In proceedings of ACM 2000 Conference on Computer Supported Cooperative Work, pp. 241–250 (2000)

[6] Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web, pp. 291–324. Springer Berlin / Heidelberg (2007)

[7] Robin Burke. Hybrid recommender systems: Survey and experiments. User modeling and user adapted interaction, 12(4):331–370, 2002.

[8] Hael Al-bashiria, Mansoor Abdullateef Abdulgabbera, Awanis Romlia, Fadhl Hujainaha. Collaborative Filtering Recommender System: Overview and Challenges

[9] Muhammad Asima, Muaaz Zakria. Advanced kNN: A Mature Machine Learning Series

[10] Joel Grus. Introduction to Data Science.

[11] Hassan Khosrav. Recommendation in Personalised Peer-Learning Environments

[12] Aleem Akhtar. Implementation of Course Recommender System for Virtual University of Pakistan.

[13] Klinkenberg S, Straatemeier M, Van der Maas HLJ. Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation // Computers & Education. — 2011. — Vol. 57, no. 2. — P. 1813–1824.
[14] Специализация Машинное обучение и анализ данных на coursera. [Электронный ресурс]. Онлайн-курсы.
URL: https://www.coursera.org/specializations/machine-learning-data-analysis
[15] Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009), no. 8, 30–37.
[16] Deerwester, S. , Dumais, S., Furnas, G.W., Landauer, T.K. and Harshman, R., “Indexing by Latent Semantic Analysis”, Journal of the Society for Information Science 41 (1990), 391– 407
[17] D. T. Pham, S. S. Dimov, and C. D. Nguyen. Selection of K in K-means clustering
[18] Pradnya Vaibhav Kulkarni. Recommender System in eLearning: A Survey.
[19] Hasan Kahtan,Hael Al-bashiri. A Proposed Course Recommender Model based on Collaborative Filtering for Course Registration